Greedy Algorithms Assignment

“A move service (driver + lorry with a load capacity of 10 Tm) has been hired by GREEDY Co. in order to carry goods from a given warehouse to another one. GC company will pay for this service at a rate of 40€ for each object transported, regardless of its weight. As there are goods with very different weights, the driver (and lorry’s owner) ask you to provide a greedy algorithm to choose the goods to be carried so that the money received from GC will be maximum. This algorithm must ask the user for the weights of the goods and after executing should show the weights of the chosen goods and the amount of the bill for the transport service.”

Main ideas

The main idea for this project is to solve it by a Greedy Algorithm. A Greedy Algorithm is described as the best way to solve any kind of problem with the most optimal solution that we can find. The code for this exercise must be effective and efficient to save as many resources as possible.

Focusing in our problem, we should find an algorithm that gives to the company GREEDY Co. the possibility to obtain the highest benefit possible when doing a transport from a warehouse to another one.

Firstly, we have to clarify who will be the candidates to be processed in our solution. As supposed, we will take the items to be transported as our candidates.

Later on, we should get into account that we will have to add fourty euros per object transported regardless of weight, so to obtain the highest benefit we need to take first the lowest objects in order to carry as many objects as possible

Finishing our explanation, it is needed to know how our code will end. In our statement we find that the maximum weight that can be transported is 10 Tons, so our summatory will end when we reach that weight, or in another words, when we get as close as possible to that weight.

Formal Description

- Candidates: Items to be transported
- Ordering: Increasing values, int i,j,k int n1=middle-f+1,n2=s-middle, int First[n1],Second[n2]
- Feasibility:

```
void merge(int arr[], int f, int middle, int s){

while(i < n1 && j < n2){
 if(F[i] <=S[j] ){
 arr[k] = F[i]
 i++
 } else {
   arr[k] = S[j];
   j++ } k++ }

while (i < n1){
arr[k] = F[i];
i++, k++;
}
```

```
while ( j < n2){
arr[k] = S[j];
j++, k++;}
}

void mergeSort(int arr[], int f, int s){
if(f < s){
int middle = f + (s - 1) / 2;
mergeSort(arr, f, middle);
mergeSort(arr, middle +1, s);
merge(arr, f, middle, s);
}}

mergeSort(arrayOfGoods, 0, numberOfObjects - 1)
```

- Selection: -
- Ending condition: (If weight + arrayOfGoods[i] ≥ 10.000)

Complexity of the algorithm

$$T(n) = 2t(n/2) + k * n \Rightarrow n = 2^i \Rightarrow T(2^i) = 2T(2^i - 1) + k * 2^i \Rightarrow (r-2)^2 \Rightarrow T(2^i) = A * 2^i + B * i * 2^i \Rightarrow T(n)$$

So, as we can see, the obtained order of the pseudocode is:

$$O(n) = n * lg(n)$$

Code explanation

First, the "merge sort" algorithm is defined in the code as a pair of functions. This algorithm divides an array by half repeatedly until only subarrays of length "one" are left. This is achieved by calling the function "Merge sort" in order to do each division.

After that, each array of length "one" is sorted individually.

Finally, each subarray is merged with another subarray of similar length. This is repeated until an array with length equal to the original array.

Following the algorithm comes the “main” function.

An array is created with index equal to the number of elements the client is going to transport, then the client enters the weight of the products to be tgransported.

A call to the previously defined functions is made in order to sort said weights and add them until the limit is reached. Following that, the price is calculated.

In the end, a list of the products along with their respective weights and the total price is displayed.

Implementation in C

```
#include <stdio.h>

void merge(double arr[], int f, int m, int s) {
        // "f" and "s" are the extreme elements of the array, defined as
        // first and second respectively
        // "m" is defined as the mid element of the array, that is why it goes through
        // "m" to "s" and then from "f" to "m + 1" (to choose the next one after the
        // mid element)
        //    F[f..........] m S[s.......]
    int i, j, k;
```

```
    int n1 = m - f + 1;
    int n2 = s - m;

        // Defines two subarrays, from "m" backwards and from "m" forwards
        // "F" is refered to the first subarray created and "S" for the second one
    double F[n1], S[n2];

        // We introduce the elements in each half of the principal array
    for (i = 0; i < n1; i++)
        F[i] = arr[f + i];
    for (j = 0; j < n2; j++)
        S[j] = arr[m + 1 + j];

        // Restart the variables to give them some use
    i = 0;
    j = 0;
    k = f;

    // Compares the elements of each sub-array to see which is lower to introduce
    // it inside first of the original array
    while (i < n1 && j < n2) {
        if (F[i] <= S[j]) {
            arr[k] = F[i];
            i++;
        } else {
            arr[k] = S[j];
            j++;
        }
        k++;
    }

        // This and the next one are simple algorithms to order the arrays
        // (or sub-arrays) related to the remaining elements
        // Copy elements from F[], if there are any left
    while (i < n1) {
        arr[k] = F[i];
        i++;
        k++;
    }
        // Copy elements from S[], if there are any left
    while (j < n2) {
        arr[k] = S[j];
        j++;
        k++;
    }
}

void mergeSort(double arr[], int f, int s) {
    if (f < s) {
        // Finds the middle point of the array
        int m = f + (s - f) / 2;

        mergeSort(arr, f, m); // Sorts 1st half
        mergeSort(arr, m + 1, s); // Sorts 2nd half
```

```
        merge(arr, f, m, s);// Merges 2 sorted halves
    }
}

int main() {
    double weight = 0;
    int numberOfObjects = 0;

    printf("Welcome to GREEDY CO. services.\nPlease, insert how many goods you want us to carry:
    scanf("%d", &numberOfObjects);

    double arrayOfGoods[numberOfObjects];

    printf("Insert the weight of said goods (In kg): \n");
    for (int i = 0; i < numberOfObjects; ++i) {
        do {
            printf("Item %d: ", i + 1);
            scanf("%lf", &arrayOfGoods[i]);
            // In case the object's weight is negative or 0
            if (arrayOfGoods[i] <= 0) {
                printf("Weight must be greater than 0. Please enter again.\n");
            }
        } while (arrayOfGoods[i] <= 0);
    }

    // Merge Sort
    mergeSort(arrayOfGoods, 0, numberOfObjects - 1);

    printf("Items taken:\n");
    int numberOfItemsAdded = 0;
    double totalPrice = 0;
    for (int i = 0; i < numberOfObjects; i++) {
        if (arrayOfGoods[i] > 0) { // Weight greater than 0
            if (weight + arrayOfGoods[i] > 10000) {
                printf("Adding another item would exceed the maximum weight of 10000 kg.\n");
                break;
            } else {
                printf("Item %d: %.2lf kg\n", i + 1, arrayOfGoods[i]);
                weight += arrayOfGoods[i];
                numberOfItemsAdded++;
                totalPrice += 40; // 40 euros per valid object
            }
        }
    }

    printf("Total weight of the goods: %.2lf kg\n", weight);
    printf("Price to pay for this service: %.2lf euros\n", totalPrice);

    return 0;
}
```

How to execute the .exe file

Once you have opened the .c file in any compiler (we will be using “Clion”) you should click on the run option.

Afterwards, our code will ask you how many goods you want the company to carry. Then you should write any value greater than cero.

```
Welcome to GREEDY CO. services.
Please, insert how many goods you want us to carry:
```

Then, you will need to introduce each product weight, again they need to be greater than 0. And our code will give you the items chosen following the greedy algorithm, the total weight of those goods and the price to pay for that service.

```
Welcome to GREEDY CO. services.
Please, insert how many goods you want us to carry:3
Insert the weight of said goods (In kg):
 Item 1:4000
 Item 2:20
 Item 3:10
 Items taken:
Item 1: 10.00 kg
Item 2: 20.00 kg
Item 3: 4000.00 kg
Total weight of the goods: 4030.00 kg
Price to pay for this service: 120.00 euros
```

Examples

Now that we have showed how to execute the code, we will show you an output where we introduce some articles without any apparent problem.

```
Welcome to GREEDY CO. services.
Please, insert how many goods you want us to carry:5
 Insert the weight of said goods (In kg):
 Item 1:3333
 Item 2:3333
 Item 3:3334
 Item 4:30
 Item 5:20
 Items taken:
Item 1: 20.00 kg
Item 2: 30.00 kg
Item 3: 3333.00 kg
Item 4: 3333.00 kg
Adding another item would exceed the maximum weight of 10000 kg.
Total weight of the goods: 6716.00 kg
Price to pay for this service: 160.00 euros
```

Now, we are showing what the outputs would be if we introduced an incorrect weight.

```
Welcome to GREEDY CO. services.
Please, insert how many goods you want us to carry:3
 Insert the weight of said goods (In kg):
Item 1:-2
 Weight must be greater than 0. Please enter again.
Item 1:0
```

```
 Weight must be greater than 0. Please enter again.
Item 1:
```

And lastly, here we have another output example where we introduce expected values so we get a expected output.

```
Welcome to GREEDY CO. services.
Please, insert how many goods you want us to carry:5
Insert the weight of said goods (In kg):
Item 1:10
 Item 2:2
 Item 3:4
 Item 4:5
 Item 5:1
 Items taken:
Item 1: 1.00 kg
Item 2: 2.00 kg
Item 3: 4.00 kg
Item 4: 5.00 kg
Item 5: 10.00 kg
Total weight of the goods: 22.00 kg
Price to pay for this service: 200.00 euros
```